

**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ**  
**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ЛЬВІВСЬКА ПОЛІТЕХНІКА**



**АВТОМАТИЗОВАНЕ ПРОЕКТУВАННЯ**  
**КОМП'ЮТЕРНИХ СИСТЕМ**

**Task2: SW <> HW**

**Виконав:**

**КІ-404**

**Копій О.Р.**

**Прийняв: Федак П.Р.**

**Львів 2024**

**Опис завдання:** В даній лабораторній роботі створити просту комунікаційну схему SW(client) <-> UART <-> HW(server), а саме виконати наступні пункти:

1. Create a simple communication schema SW(client) <-> UART <-> HW(server).
2. The client should send a message to the server. The server should modify the message and send it back to the client.
3. Create YML file with next features:
  - a. build all binaries (create scripts in folder ci/ if need);
  - b. run tests;
  - c. create artifacts with binaries and test reports;
4. Required steps

### **Теоретичні відомості:**

В основі клієнт-серверної архітектури лежать два компоненти: клієнт і сервер.

**Клієнт** – комп'ютер на стороні користувача, який відправляє запит до сервера для надання інформації або виконання певних дій.

**Сервер** – більш потужний комп'ютер або обладнання, призначене для вирішення певних завдань з виконання програмних кодів, виконання сервісних функцій за запитом клієнтів, надання користувачам доступу до певних ресурсів, зберігання інформації і баз даних.

Модель такої системи полягає в тому, що клієнт відправляє запит на сервер, де він обробляється, і готовий результат відправляється клієнтові. Сервер може обслуговувати кілька клієнтів одночасно. Якщо одночасно приходить більше одного запиту, то вони встановлюються в чергу і виконуються сервером послідовно. Іноді запити можуть мати пріоритети. Запити з більш високими пріоритетами повинні виконуватися раніше.

**UART** — тип асинхронного приймача-передавача, компонентів комп'ютерів та периферійних пристроїв, що передає дані між паралельною та послідовною формами.

Біти даних передаються з одного місця в інше через дроти або інші носії. Якщо мова йде про великі відстані, вартість дротів стає великою. Щоб зменшити вартість довгих комунікацій, що переносять кілька біт паралельно, біти даних передають послідовно один за одним, і використовують UART для перетворення паралельної форми на послідовну на кожному кінці лінії зв'язку. Кожен UART має зсувний регістр, який є фундаментальним методом для перетворення між паралельними та послідовними формами.

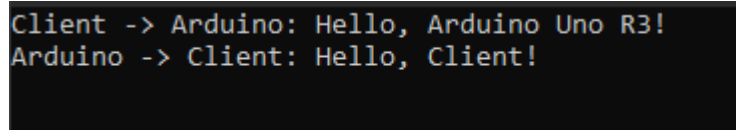
Зазвичай UART не отримує і не генерує зовнішні сигнали, які подорожують між різними частинами обладнання. Як правило, для перетворення логічного рівня UART в та з зовнішнього рівня сигналів використовується окремий інтерфейсний блок.

**YML** (YAML) файл у контексті GitHub Actions використовується для опису конфігурацій робочих процесів (workflows) для автоматизації завдань, таких як збірка, тестування, розгортання додатків тощо. GitHub Actions дозволяє запускати ці процеси автоматично на основі певних подій (наприклад, push, pull request) або за розкладом.

### Деталі реалізації:

Було виконано наступні кроки для реалізації завдання:

- Створено нову гілку feature/develop/task2 на Github;
- Розроблено просту комунікаційну схему згідно завдання:
  - Початковий код клієнта був написаний мовою C++, у подальшому до котрої буде приєднуватись бібліотека Qt – крос-платформовий інструментарій, котрий в даному випадку буде допомагати застосовувати графічні елементи у нашому додатку.
  - Початковий код сервера був написаний у програмі Arduino IDE. У подальшому саме він буде відповідати на всі запити клієнта.
  - Комунікація UART була здійснена через порт COM3. Саме через нього Arduino UNO R3 та клієнт будуть комунікувати в подальшому.



```
Client -> Arduino: Hello, Arduino Uno R3!  
Arduino -> Client: Hello, Client!
```

*Рис.1. Взаємодія SW <-> UART <-> HW.*

- Розроблено YML файл для побудови бінарних файлів серверної та клієнтської частини програми. Згідно до вимог завдання, він також проводив тестування нещодавно створених бінарних файлів, а також створював артефакти з бінарними файлами та результатами тестів:

14 workflow runs				Event ▾	Status ▾	Branch ▾	Actor ▾
✓ 14	Build Project #16: Commit <code>ff4e20d</code> pushed by SamuraiSanch	<code>feature/develop/task2</code>	2 days ago 14m 43s	...			
✓ 13	Build Project #15: Commit <code>96246cc</code> pushed by SamuraiSanch	<code>feature/develop/task2</code>	2 days ago 14m 59s	...			
✓ 12	Build Project #14: Commit <code>b1eba5a</code> pushed by SamuraiSanch	<code>feature/develop/task2</code>	2 days ago 14m 40s	...			

*Рис.2. Тестування коректної роботи YML файлу.*

Artifacts			
Produced during runtime			
Name	Size		
📁 arduino_binaries	78.4 KB	📄	🗑
📁 vs_binaries	1.42 MB	📄	🗑
📁 vs_executable_result	32 Bytes	📄	🗑

*Рис.3. Результат виконання Github Actions завдяки розробленому YML файлу.*

- У останньому пункті було виконано всі необхідні кроки для кожної лабораторної роботи, згідно методичних вказівок.

**Висновок:** На даній лабораторній роботі я створив нову гілку в своєму репозиторії на Github, створив файли згідно завдання та завантажив туди, а саме: клієнт, котрий комунікує з сервером завдяки порту COM3. Також створив та завантажив YML файл, що буде файли серверної та клієнтської частини, проводить тестування та створює артефакти на основі отриманих результатів.

### **Списки використаної літератури:**

1. Laster, B. *Learning GitHub Actions: Automation and Integration of CI/CD with GitHub*. 1st ed., 411 pages, 2023.
2. Yādava, S.C. *Introduction to Client Server Computing*. Hardcover, 544 pages, 2009.
3. [https://en.wikipedia.org/wiki/Client%E2%80%93server\\_model](https://en.wikipedia.org/wiki/Client%E2%80%93server_model)